SECURE COMPUTER SYSTEMS

HOME WORK – 1

Submitted By: Manish Choudhary

Answer 1

a) C2 and B1

B1 provides all the features covered in C2 and also introduces some new design/policy requirements in addition. The primary distinguishing feature is the introduction of mandatory access control over selected subjects and objects. So, it affects the functionality, design and implementation considerations.

If a system provides discretionary access control only, it will fall under C1 or C2 depending upon the other features while if the system ensures selected mandatory protection, it can be considered for level B1.

In addition to the above mentioned distinguishing feature, B1 also requires:

- Data labeling based on the sensitivity level and capabilities to export them
- Some design specification and verification (thus, adding some testing/verification feature)
- Informal security policy model

But, the key difference is the MAC.

Windows NT Server and Workstation 4.0 were given C2 level clearance in 1999. At that time, C2 was considered to be the highest evaluation rating that can be achieved by a general-purpose operating system. Trusted Solaris V1.1 and HP-UX BLS, 9.09+ are examples of B1 certified systems.

b) B1 and A1

A1 requires the system to meet all the requirements mentioned for level B1. In addition, it should provide the features as stated in B2, B3 and some other mentioned in A1. Although, B2 and B3 also add many requirements but, the primary distinguishing feature between B1 and A1 is the introduction of verified protection. To get A1 level clearance, the system must provide formal design and verification techniques. So, the verification requirements are very high and must be met using some formal methodology. Formal top level specification must be provided.

In addition, A1 requires (just the key additions):

- DAC and MAC for all objects while it is for selected objects in B1 [added in B2]
- layered design of the system with security domains which requires changes in design and implementation [added in B3]
- Formal and properly defined security policy model [added in B2]

The key difference is the addition of the requirement of formal design verification. The correctness of the implementation is equally important and can't be ignored.

Honeywell SCOMP, USAF SACDIN, NSA Blacker and Boeing MLS LAN are some examples of A1 certified systems.

c) EAL4+ Certified Oracle Solaris 11.1 Operating System

EAL stands for Evaluation Assurance Level, a numerical rating to describe the depth and rigor of an evaluation of a system. This is a rating methodology used in **Common Criteria for Information Technology Security Evaluation standard**, also known as **Common Criteria**. There are 7 levels starting with EAL1 (the simplest one). EAL4 is the highest level achievable for commercial software. EAL4+ refers to EAL 4 augmented evaluation.

EAL4 requires the system to be methodically designed, tested and reviewed in addition to the normal functional implementations specified in the Protection Profile. It mandates the design description and documentation as well as regressive in-depth testing of the system. The levels above this add the need for semiformal or formal design and verification. Therefore, EAL4 (or EAL4+) is the highest level that most of the practical systems are able to achieve. Here, evaluation is carried out based on some specific configurations mentioned by the vendor and thus, is allowed to make certain assumptions about the threat landscape.

Oracle Solaris OS has achieved EAL4+ level certification on March 18, 2014. It is **conformant to the BSI Operating System Protection Profile v2.0** with following extended packages:

- Advanced Management
- Extended identification and Authentication
- Labeled Security
- Virtualization

It also fulfills the functional as well as the security assurance requirements. ALC_FLR.3 – Systematic Flaw Remediation is an augmented feature.

The functional requirements of common criteria can be expressed in terms of classes, families and components. The various functional requirement classes and how Solaris satisfied them are described below. It is just an abstract level and the detailed description can be found in the references provided (after this answer):

- **Security Audit (recognizing, recording, storing, and analyzing information related to security relevant activities):** The system is capable of collecting extensive audit information about security related actions taken or attempted by the users making them accountable. It allows selective auditing, alarm generation and notification, administrative control and remote storage of audit file.
- **Cryptographic Support (cryptographic functionality to help satisfy several high-level security objectives):** Solaris supports FIPS-validated cryptographic modules to provide cryptographic

functionality for authentication, data protection, and communications security purposes. It includes symmetric, asymmetric, hashing, random number generation, and message authentication code algorithms which are implemented via multiple providers plug-in.

- **User Data Protection (families specifying requirements related to protecting user data):** Following policies have been provided for data and information flow protection. Persistent Storage Object Access Control Policy, Transient Storage Object Access Control Policy, Zone Access Control Policy, Network Information Flow Control Policy, Multilevel Confidentiality Information Flow Control Policy, and Zone Information Flow Control Policy
- **Identification and Authentication (requirements for functions to establish and verify a claimed user identity.):** Solaris provides the functionality to enforce the identification and authentication of the users by maintaining security attributes such as User Id, password, security roles, certificates, privileges etc. for each user. It also provides protected authentication feedback, enforces minimum password strength requirements and fail safe login. It provides multiple authentication mechanisms based on passwords, tokens, certificates, Kerberos tickets and public key user authentication. It also supports remote login.
- **Security Management (management of several aspects of the TSF such as security attributes, TSF data and functions):** Solaris implements role based access control policy to selectively grant administrative rights to users or roles as needed. It also supports multiple administrators with distributed admin rights.
- **Protection of the TOE* Security Functionality (TSF)(requirements that relate to the integrity and management of the mechanisms that constitute the TSF):** Solaris provides functionality for reliable time stamps, consistent interpretation of access control and information flow control related security attributes as well as label related security attributes.
- **TOE* Access(requirements for controlling the establishment of a user's session):** Solaris mitigates unauthorized user access by automatically locking a user session after an administrator predefined time interval of inactivity, allows user-initiated locking and re-authentication after timeout.
- **Trusted Path/Channels (requirements for a trusted communication path between users and the TSF and for a trusted communication channel between the TSF and other trusted IT products):** Solaris implements cryptographically protected communication channels between itself and another IT trusted products.

Other important features are: Simplified administration, designed-in virtualization, scalable data management, Compliance Reporting & SCAP Support, Extended Application Security Policies, Enhanced Exploit Mitigation (ASLR and Supervised mode execution prevention).

Augmented Flaw Remediation: Procedures to track and correct security flaws, and distribute the flaw information and corrections to users of the system

Evaluation involved (not a complete list) following:

- Functional specification and design documentation
- Ensured that initialization process is secure

- Security functions are protected against tamper and bypass
- Secure implementation of security domains.
- Verification that the design documentation, preparative user guidance and operational user guidance are complete and sufficiently detailed to result in a secure configuration
- Verification that the configuration management system and associated documentation ware mature and well-developed.
- Description of procedures required to maintain the integrity in the delivery document.

Important types are evaluation tasks performed:

- Assessing developer tests
- Performing independent functional tests (Repeat of Developer's Tests, Secure Communication, Label usage, Access restrictions)
- Performing penetration tests

The Solaris 11.1 couldn't go beyond this level as it doesn't provide semi-formal or formal verification methodologies which are the basic requirement for the levels above EAL4.

References:

https://www.cse-cst.gc.ca/en/system/files/pdf_documents/oracle-solaris-v111-sec-eng.pdf

http://www.oracle.com/technetwork/topics/security/oracle-solaris11-1-cr-2179650.pdf

http://www.commoncriteriaportal.org/files/ccfiles/CCPART2V3.1R2.pdf

*Target of Evaluation

Answer 2

a) Location of RPL Bits

RPL bits are the bits 0 and 1 in the segment selector, a 16 bit identifier for each segment which points to the segment descriptor that defines the segment.

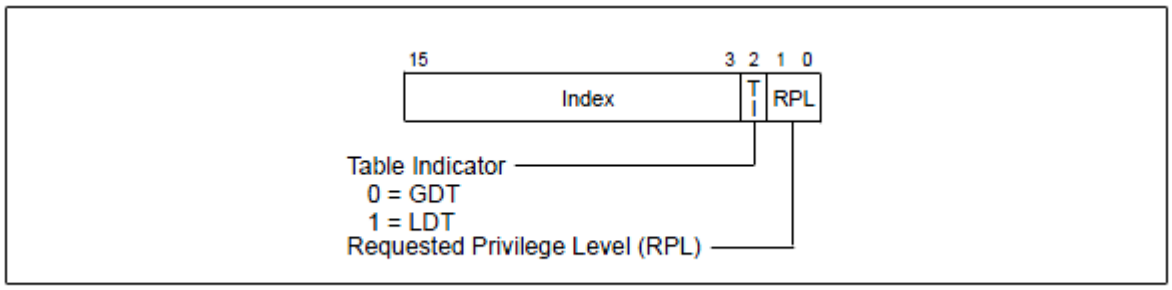

Figure 3-6. Segment Selector

(The figure has been borrowed from the Intel manual)

b) Need for RPL Bits

RPL bits specify the privilege level assigned to a segment selector which can range from 0 to 3. It is an override privilege level which is used to insure that a privileged code can't access a segment on behalf of an application unless this application process itself has the access privilege for that segment. It is done by carrying the current privilege level of a calling procedure to the called procedure. The calling procedure passes the segment selector (for the data segment to be accessed) to the called procedure. Before passing the segment selector, the calling procedure sets the RPL bits (in the segment selector) equal to its CPL. After that, RPL is used by the called procedure to determine the access privileges to a segment.

In general, the RPL and CPL should be numerically smaller than or equal to the DPL of the segment containing the data to allow access. In case of a stack segment, all the three privilege levels (RPL, CPL and DPL) should be equal.

A common usage is by the OS procedures to prevent application procedures from accessing data which can't be accessed directly by the application procedure. Thus, RPL bits can help in preventing the accidental (or intended) use of segment selectors for privileged data segments by less privileged programs or procedures.

To detect the wrong RPL bits passed by the caller (the application program), the operating system can use the ARPL instruction to check if the caller's CPL (from the code-segment selector pushed on the stack) is same as the RPL bits passed in the data segment selector.

c) Check for RPL Bits

Let's say an application procedure is executing with some low privilege level and it wants to access some high privilege level data through another privileged procedure. The application procedure would pass the logical address for the data to the called procedure. The segment selector for the segment to be accessed is a part of this argument passed to the called procedure. When the callee wants to access the data in the segment, its segment selector gets loaded in the segment register. But, before loading it into the segment register, the processor checks that the RPL bits in the segment selector and the CPL of the called procedure are numerically less than or equal (or just equal to in case of stack segment) to the DPL bits in the segment descriptor of the data segment. The segment descriptor is accessed using the index in the segment selector. This index is multiplied by 8 and added to the base address of GDT or LDT (stored in GDTR or LDTR) which gives the base address of 8 bytes long segment descriptor. The DPL bits in this descriptor are checked against RPL and CPL (the bits 0 and 1 in the CS segment register). If the privilege is verified, other information from the segment descriptor is loaded in the segment register after additional checks are performed. After that, the base address present in the segment descriptor is added to the offset present in the logical address to access the data in the segment. If paging has been enabled, this linear address in converted to the physical address using the hierarchical page table structure.

If an application program running at CPL3 wants to access data segment with DPL 0, it makes a call to the operating system via a call gate passing the segment selector for the data segment on a stack after setting its RPL to 3. Now, if the OS tries to access the data segment using the segment selector (which is actually a part of the virtual address) passed on the stack, the processor compares the CPL (which is 0 now), RPL of the segment selector and DPL of data segment. It would result in an access denied as the RPL is numerically greater than the DPL. This results in a protection fault.

Other cases where fault would occur because of the RPL bits:

- CPL=0/1/2, RPL = 3, DPL = 2
- CPL = 0/1, RPL = 2, DPL = 1
- CPL=0, RPL = 1, DPL =0
- !(CPL=RPL=DPL), in case of stack segment

So, whenever the RPL is greater than the DPL (even though CPL is less than or equal to DPL), it would result in a fault. In case of a stack segment, whenever the RPL is not equal to DPL (even though CPL is equal to DPL), it would result in a fault.

d) SLDT and INVLPG

SLDT is not protected from the application while INVLPG is protected.
SLDT: Stores the LDT segment selector from the LDTR register into memory or a general-purpose register. This instruction can be executed in protected mode.
INVLPG: flushes the TLB entry specified with the source operand. It is a privileged instruction. CPL of a program must be 0 to execute it.

Storing the LDTR value doesn't affect the other processes and thus, it is allowed even for the applications. This instruction doesn't affect any of the state flags. The value stored may be used by the application in some way. But, it is not interfering with the execution of other application processes running on the same machine. Therefore, writing to this register is a privileged operation (as it can have impact on other processes) but reading the value from the register (SLDT instruction) is not privileged one.
The applications normally have no use of this instruction and thus, could have been made privileged.

INVLPG flushes the TLB entry. If allowed, a process can flush any of the TLB entries affecting other processes and thus, should be accessed at highest privilege level only and should be protected from any user level application. If the TLB is tagged, it will have mapping for different processes as it helps us in avoiding TLB flush at every context switch. If the user application is allowed to execute this instruction, it can flush mapping for other processes and thus, affecting their execution.

INVLPG is already a privileged instruction and thus, doesn't impact virtualization as all the privileged instructions are trapped to the VMM. Therefore, it needs to be privileged for virtualization and it is already privileged.

SLDT is not a privileged instruction. This is a sensitive instruction from Virtualization perspective but it doesn't trap to the VMM.
If a guest VM executes this instruction, it will get the value associated with the LDTR applicable to the host. This can be problematic if the value is used by the guest VM in some way as the values conflict. Therefore, if host and guest are running together, we need to relocate LDTR as there is only one LDTR actually. This can be exploited by an application running on the guest VM to know that it is running in a virtual environment. This technique was a common one employed by the malware authors to detect the presence of virtual environment. SLDT should have been a privileged instruction to allow transparent virtualization but it is not. If VMX technology is used, we don't need to worry about this as the hardware provides the protection.

Answer 3

a) If the target page is in memory, the guest OS can handle this operation independently without the intervention of the hypervisor. It is because the guest OS has the read access to the hardware pages containing page tables and thus can perform the address translation. But, if there was a page fault (target page not present in the memory), it had to update the page table for which hypervisor had to be invoked as the guest doesn't have the right to modify the page tables.

b) Hardware interrupts are delivered to the guest OS through an asynchronous event based mechanism which requires hypervisor's intervention. Important notifications are delivered as events. Xen updates the pending events and delivers them by invoking an event-callback handler specified by the guest OS. Therefore, hardware interrupts are virtualized by mapping them to the events. Virtual IDTs contains the handlers corresponding to these events.

c) Creating a VM requires the intervention of the hypervisor. Creating VM requires virtual RAM, Disk, devices etc. which affects the state of the whole system. Only hypervisor can perform this task. The initial domain created at the boot time (Domain0) is given permission to use control interface. The other VMs can be created using this control interface. Therefore, the guest VM is not capable of performing VM creation task independently. The Dom0 makes some hyper calls to the hypervisor to configure the initial allocation of space to a domain, configure the initial page tables etc.

d) There is no need to invoke hypervisor to perform system calls made from the application level. Each guest OS registers a fast exception handler which is accessed directly by the processor without in-directing via the hypervisor. Xen hypervisor needs to validate this handler only once before installing it in the hardware exception table. After that, system calls can be made directly through this exception handler.

e) The main motivation for introducing VMX operation is to facilitate hardware based virtualization. The support from the hardware simplifies the development of virtual machine monitors and gives better control to them. VMX operation is of two types, VMX root operation and VMX non-root operation. VMM runs in VMX root operation and the guest runs in VMX non-root operation. The hardware helps to restrict the possible functions in VMX non-root operation giving control to the VMM running in VMX root operation on execution of certain instructions. This gives more control to the VMM and also allows the guest software to run with its original privilege level (for which it was designed) eliminating the need for de-privileging of ring for the guest OS and thus, reducing the VMM's pain (virtualization overhead and performance).

f) VM_EXIT instruction is to transit from the VMX non-root operation (where guest is running) to the VMX root operation (where VMM is running). This gives control to the VMM. VM exit is

caused on execution of certain instructions (page faults etc.) by the guest. VM_EXIT transfers the control to the entry point specified by the VMM. It involves recording the information about the cause in the VM-exit information field and also requires saving a snapshot of the VM's state. This is done by updating the VM control structure. The registers and the address space are swapped in one atomic operation. After that, VMM can analyze the cause for VM Exit and can take suitable action. Finally, control can be transferred back to the guest by executing VM_ENTRY instruction.

Answer 4

a) e1+e2 (The entropy gets added in case we mandate both the authentication methods)

b) Min (e1, e2). In case any one of the method would allow access, the attack would have to just defeat the method with lesser entropy (as it would be easier for him to guess/ brute force the method with lesser entropy).

c) No, the two factors are not independent as per the definition given in the question. If the keystroke dynamics factor is compromised, the adversary would have the instruction table. The total number of entries in the table tells the number of features which in result would leak the length of the password. For example, if the total number of features is 15, we know that the password length is 8(8 key presses and 7 key transitions). This is possible assuming that the adversary knows the basis of our feature selection and this is generally true as we can't base the security of our system on hiding such information.
Therefore, adversary has an advantage to compromise a factor if he has compromised the other.

d) Blinding can be performed. Instead of using (m, m) secret sharing scheme, we can use (k, m) scheme. Here k would be the actual number of features and m would be a fixed parameter. As the values in the two columns are random, we can update the values for first k features depending upon the measurements and the rest can be filled with pseudo random values.
The first k rows can be used to retrieve the hardened password and value of k can be determined at run time on the basis of input password. In case of a compromise, adversary would always see m rows in the table for any user account and thus, wouldn't be able to predict the length of the password.
Drawbacks: 1) It requires more storage as we have to choose sufficiently high value of m such that it can handle different lengths of passwords.
2) We may have to enforce an upper limit on the number of characters if we want to control the storage requirement.
But, given that storage is really cheap now, we can have a large value for m.

e) Minimum: If all the features are distinguishing, all the features would contribute to the entropy. The worst possible case would be when all the features fall in the same column (all are either fast or slow). In that scenario, adversary has to just guess the correct column. If the entropy of the password is k, total entropy would be **$k + \log_2(2)$** (as the work factor would be 2 and thus entropy would be $\log_2(2)$ or 1).

Maximum Entropy: **k + m** where m is the total number of features (as all are distinguishing). The best case for us would be when all the features fall in slow or fast column randomly and

thus, the adversary would have to try all possible combinations. Total such possible combinations would be 2^m.

This can be explained with simple permutation and combination.

In case of maximum entropy, each feature may fall either in left or in right column. So, for each feature, adversary would have to guess whether the right value falls in left or right column. So, the total such possible options will be 2x2x2…(m times). Therefore, he may have to attempt 2^m times to get the right one. Hence, the entropy would be base-2 log of 2^m which is m.

The minimum entropy case would be when adversary has to guess the correct column for just one feature and all other features would fall in the same column. Therefore, the work required will be 2x1x1…(total m-1 1's) i.e., 2. Hence the entropy would be base-2 log of 2 which is 1.